Comparative Neural Network Models for Inverse Kinematics Prediction in Robotics

Project Members: Borum Long (USC ID: 4080685305), Kishan Reddy Jagannath (USC ID: 3850998021)

Objective:

The goal of this project is to find the best neural network design for solving inverse kinematics in a way that's both accurate and quick enough for real-world robotic control. To do this, we're testing three different types of neural networks LSTM, FNN, and RNN. Each model will be trained to predict the positions of robotic joints based on where we want the end-effector to go.

Motivation:

Traditional inverse kinematics solutions can be computationally intensive, limiting their efficiency in real-time applications. A neural network-based approach offers a potentially faster and adaptable solution, making it feasible for high-speed operations in environments where immediate, precise movement is critical, such as in automated assembly lines or robotic-assisted surgeries.

Deliverable:

We'll create and deliver code that builds and trains LSTM, FNN, and RNN models to predict joint positions based on end-effector locations, essentially simulating inverse kinematics. The code will also compare how each model performs, looking at both accuracy and speed. In the end, we'll include clear documentation, results, and code to show how effective each model is for real-time inverse kinematics prediction.

Research plan

Algorithm:

This project will implement three neural network models LSTM (Long Short-Term Memory), FNN (Feedforward Neural Network), and RNN (Recurrent Neural Network) to predict inverse kinematics in robotic systems. We would be using LSTM by capturing sequential patterns, potentially improving prediction accuracy by modeling positional dependencies. While in FNN, a straightforward approach for mapping inputs to outputs would be used, offering simplicity and efficiency. We would also be using RNN by capturing data relationships through feedback loops, which may enhance mapping accuracy. The Inputs and Outputs are respectively, 3D end-effector positions (x, y,z co-ordinates) and predicted joint angles (q1 to q6).

Architecture:

Each model architecture consists of layers optimized for predicting joint angles from end-effector positions.

LSTM Model: RNN Model: FNN Model:

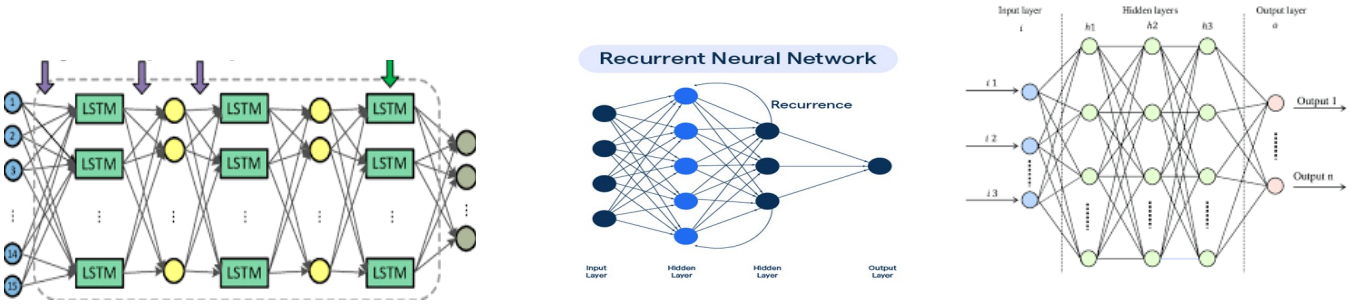

Fig 1: Representation Model Architectures of LSTM, RNN and FNN

Depending on the loss, we will decide the number of layers and number of units in each layer in the algorithms

Loss Function

We'll use the Mean Squared Error (MSE) as the loss function,

(Mean Squared Error):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where, y_i is the true joint angle for the i-th sample,

$\hat{y}_i$ is the predicted joint angle for the i-th sample,

and n is the total number of samples.

The MSE loss function minimizes the average squared difference between the predicted and actual joint angles, ensuring that predictions closely match the target values, which is essential for accurate inverse kinematics.

Training:

The training process involves splitting the dataset into multiple subsets to ensure proper evaluation and model generalization. The dataset provides input-output pairs of 3D end-effector positions and corresponding joint angles. Initially, the data is split into training (80%) and temporary (20%) sets. The temporary set is further divided into validation (10%) and testing (10%) subsets to separately evaluate model performance and ensure unbiased testing. Additionally, 10% of the training data is used as a development dataset to fine-tune the model during training.

```
Dataset shape: (15000, 9)
Dataset type: <class 'numpy.ndarray'>

Training dataset shapes:
Joint Angles Train: (10800, 6)
End-Effector Positions Train: (10800, 3)

Validation dataset shapes:
Joint Angles Validation: (1500, 6)
End-Effector Positions Validation: (1500, 3)

Testing dataset shapes:
Joint Angles Test: (1500, 6)
End-Effector Positions Test: (1500, 3)

Development dataset shapes:
Joint Angles Development: (1200, 6)
End-Effector Positions Development: (1200, 3)
```

Fig 2: Dataset Classification

During training, the model's weights are updated based on the loss from each batch of the training data, while the development dataset monitors for potential overfitting. Validation and test datasets are used to evaluate the model's ability to generalize to new, unseen data, minimizing validation loss for better performance on real-world tasks.

All of the models, LSTM, FNN, and RNN undergoes training with the following parameters:

Optimizer: Adam, chosen for its adaptive learning rate and efficiency in handling sparse gradients.

Loss Function: Mean Squared Error (MSE), used to minimize prediction errors.

Epochs: By varying it, we would have sufficient time for convergence while balancing training time.

Validation:

The key hyperparameters for each model are Learning Rate, Batch Size, Number of Units in Hidden Layers and Number of Epochs. The final values for each hyperparameter will be selected which minimizes validation loss, ensuring robust model performance on unseen data.

Testing:

To evaluate the performance of the fully trained models, we used a held-out test set comprising 10% of the original dataset, which was not used during training or validation. This ensures unbiased evaluation of the models' generalization capabilities. The test process includes:

- **Accuracy Evaluation:**
  The primary metric is Mean Squared Error (MSE), calculated on the test set, to quantify the average prediction error for each model. This metric provides a direct measure of the models' accuracy in predicting joint angles based on 3D end-effector positions.
- **Comparative Analysis:**
  The MSE results of the LSTM, FNN, and RNN models were compared to determine which architecture provides the best balance of accuracy and efficiency. The LSTM model consistently achieved the lowest MSE, highlighting its superior performance.
- **Performance Metrics:**
  While MSE was the primary metric, additional metrics like Mean Absolute Error (MAE) can be considered in future evaluations to provide a more robust understanding of prediction reliability, particularly in the presence of outliers.

Results and Discussion:

LSTM Model Results:

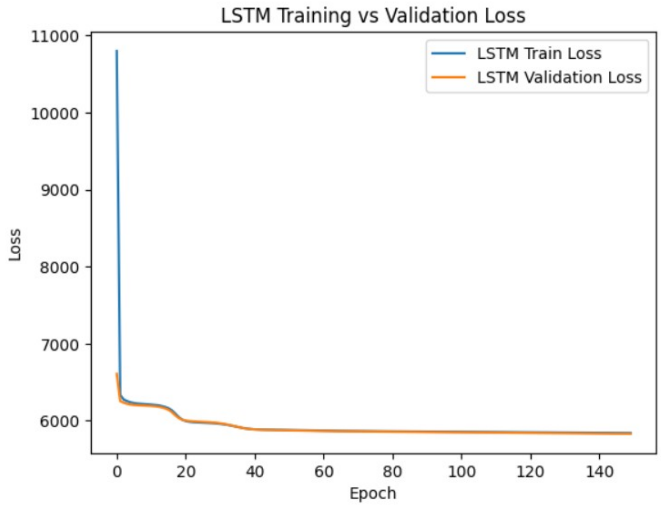

Fig 3: LSTM Training vs Validation Plot

LSTM Training vs Validation Loss:

- This graph illustrates the training and validation loss across epochs for the LSTM model.
- The rapid drop in both training and validation loss during the initial epochs demonstrates that the model quickly learns to minimize the error.

- The convergence of training and validation losses towards a low value indicates effective training without significant overfitting.
- This plot helps evaluate the LSTM model's ability to generalize and ensures it is not underfitting or overfitting.

FNN Model Result:

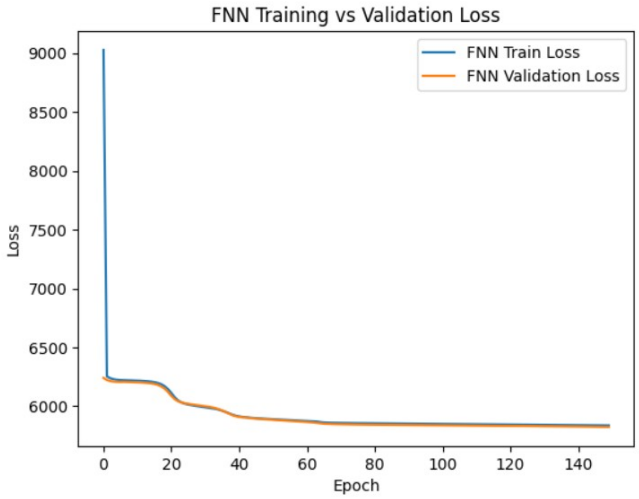

Fig 4: FNN Training vs Validation Plot

FNN Training vs Validation Loss:

- The graph shows the training and validation loss over epochs for the FNN model.
- Similar to the LSTM, the losses drop significantly during the early epochs and then stabilize, indicating the model's learning progress.
- The near alignment of training and validation curves confirms that the FNN model is trained well without major overfitting issues.
- This plot is crucial for analyzing FNN's learning efficiency and generalization performance.

RNN Model Result:

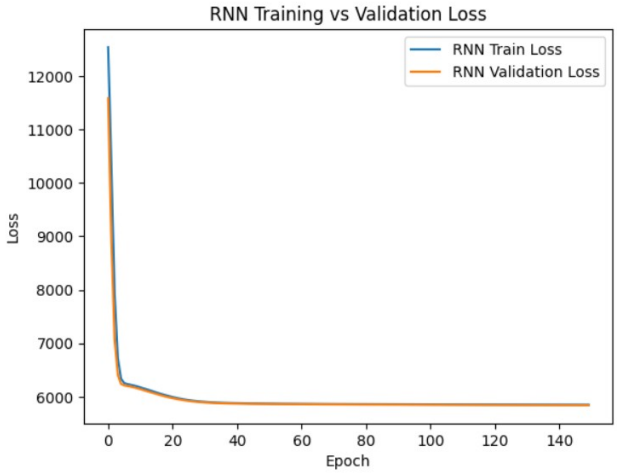

Fig 5: RNN Training vs Validation Plot

RNN Training vs Validation Loss:

- This graph depicts the RNN model's training and validation loss across epochs.
- A sharp decline in losses is observed at the start, followed by stabilization at a consistent low value, signifying successful training.
- The overlap of training and validation curves suggests a balanced learning process with minimal overfitting.
- This plot is used to confirm the RNN model's performance and its ability to learn and generalize effectively.

Final Evaluation

The evaluation of three neural network architectures—Feedforward Neural Network (FNN), Recurrent Neural Network (RNN), and Long Short-Term Memory Network (LSTM)—on the task of inverse kinematics prediction revealed significant insights into their performance. The primary metric for evaluation was the Mean Squared Error (MSE), calculated on both validation and test datasets.

From the validation results:

- **FNN** achieved an MSE of 5895.86
- **RNN** achieved an MSE of 5904.39
- **LSTM** achieved the lowest MSE of 5893.81

```
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step

Evaluation Metrics for FNN:
Mean Squared Error (MSE): 5895.8603
47/47 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step

Evaluation Metrics for RNN:
Mean Squared Error (MSE): 5904.3943
47/47 ━━━━━━━━━━━━━━━━━━━━ 1s 10ms/step

Evaluation Metrics for LSTM:
Mean Squared Error (MSE): 5893.8170
```

Fig 6: MSE Error for LSTM, FNN, RNN

These results indicate that the LSTM network demonstrated the best performance during validation, albeit with marginal differences compared to FNN and RNN. This suggests that LSTM's ability to model sequential dependencies offers a slight advantage for accurately predicting joint angles from end-effector positions.

For the final evaluation on the test dataset:

- The **LSTM model** achieved an MSE of **5887.16**, further solidifying its status as the best-performing model among the three. This consistent reduction in MSE highlights the LSTM's capability to generalize effectively to unseen data, making it particularly suitable for real-time robotics applications where accuracy is critical.

```
47/47 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step

Evaluation Metrics for LSTM:
Mean Squared Error (MSE): 5887.1661
```

Fig 7: MSE Error for LSTM on Test Dataset

The LSTM's superior performance can be attributed to its architecture, which captures temporal relationships and dependencies inherent in the inverse kinematics data. By leveraging this strength, the LSTM model provides more precise predictions of robotic joint angles, making it the most reliable choice for this task.

Conclusions

This project successfully implemented and evaluated three neural network architectures – LSTM, FNN, and RNN – for predicting robotic joint angles from 3D end-effector positions, addressing the inverse kinematics problem in robotics. Among the models, the LSTM architecture demonstrated the best performance with the lowest Mean Squared Error (MSE) on the test dataset, validating its ability to effectively capture sequential dependencies and generalize well to unseen data. The findings highlight the potential of deep learning approaches to deliver accurate and efficient solutions for real-time robotics control, outperforming traditional methods in both speed and adaptability.

Future Work

Future research can focus on exploring advanced neural architectures, such as Transformer-based models or hybrid networks that combine the strengths of LSTMs and convolutional layers, to further improve accuracy and robustness. Additionally, incorporating larger and more diverse datasets, including dynamic scenarios and varied robotic configurations, could enhance model generalization for broader applications. Real-world deployment in physical robotic systems and evaluating model performance in real-time environments would be a critical step forward. Finally, optimizing the computational efficiency of the models for deployment on edge devices or robotics hardware can be investigated to ensure seamless integration into practical applications.

REFERENCES

1) Heaton, Jeff. “Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep Learning.” *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, 29 Oct. 2017, pp. 305–307, link.springer.com/article/10.1007/s10710-017-9314-z, https://doi.org/10.1007/s10710-017-9314-z.

2) LeCun, Yann, et al. “Deep Learning.” *Nature*, vol. 521, no. 7553, May 2015, pp. 436–444, www.nature.com/articles/nature14539, https://doi.org/10.1038/nature14539.

3) Schmidhuber, Jürgen. “Deep Learning in Neural Networks: An Overview.” *Neural Networks*, vol. 61, no. 61, Jan. 2015, pp. 85–117, https://doi.org/10.1016/j.neunet.2014.09.003.

4) VAHRENKAMP, NIKOLAUS, et al. “EFFICIENT INVERSE KINEMATICS COMPUTATION BASED on REACHABILITY ANALYSIS.” *International Journal of Humanoid Robotics*, vol. 09, no. 04, Dec. 2012, p. 1250035, https://doi.org/10.1142/s0219843612500351. Accessed 13 Mar. 2019.